Adaptive Resource Management Technology for NASA Computing Systems

Lonnie R Welch and David M Chelberg
Center for Intelligent, Distributed & Dependable Systems
Ohio University, Athens, OH 45701
{welch | chelberg}@ohio.edu

Barb Pfarr
Systems Engineering and Advanced Concepts Division
NASA Goddard Space Flight Center, Greenbelt, MD 20771
barbara.b.pfarr@nasa.gov

***Abstract* - This paper addresses the problem of making the best use of computing and buffer resources in systems that operate in dynamic environments. The problem is addressed by providing solutions for system specification, performance monitoring, allocation optimization, and allocation enactment. This paper presents these solutions and shows how they have been applied to a prototype of a potential satellite system of the near future.**

I. Introduction

NASA satellite and robot systems operate in dynamic environments, wherein the conditions cannot be known a priori. Thus, the ability for these systems to adapt dynamically is critical. If they cannot adapt, then the return on investment for these systems is likely to be much less than could be achieved otherwise. Furthermore, autonomous adaptation is desirable, due to the complexities involved in determining appropriate adaptations and because of the high latencies in commanding the adaptations from the earth.

This paper addresses the problem of making the best use of computing and buffer resources for systems that operate in dynamic environments. To understand the problem, consider current earth-observing satellites, which have the following properties:

- Software processes are allocated to computing and storage resources statically.
- Collected scientific data is stored in a buffer and downloaded in first-in-first-out (FIFO) order.
- Algorithms are designed to have only a single version that runs at a fixed fidelity.
- Buffer overflow is handled by simply discarding any scientific data that is collected after the buffer is full, even though the new data may be more valuable than data that is already stored in the buffer.

This work was funded in part by the NASA Earth Science Technology Office Advanced Information Systems Technology Program; and by the NASA Computing, Information and Communications Technology Program.

This approach has several drawbacks. Operations costs are high, because satellites require full-time ground support to monitor and maintain satellite health, and engineers are required to solve resource allocation anomalies that may occur onboard. The benefit obtained from satellite operations is sometimes sub-optimal, because valuable data may be discarded when a buffer is full of less valuable data. Furthermore, the system has limited capabilities for graceful degradation when computing and storage resources become overloaded due to anomalies. Finally, errors and allocation anomalies may persist for prolonged periods of time because monitoring and maintenance of satellite health and safety data is performed offline.

This paper shows how to address these drawbacks with adaptive resource management middleware that operates online and allocates computing and storage resources optimally. The technology is useful for onboard satellite systems as well as for robotic systems. Specifically, it will (1) enable satellites to autonomously decide which data should be captured and stored, and (2) allow robots to determine dynamically how to best use their computing resources, to plan, to analyze, to explore, etc.

The problem is addressed by providing solutions for system specification, performance monitoring, allocation optimization, and allocation enactment. The remainder of this paper presents these solutions and shows how they have been applied to a prototype of a futuristic satellite system.

II. Resource Manager

In previous years, the authors of this paper have developed adaptive resource management [1] middleware solutions primarily in the context of dynamic shipboard computing [2]. Recent efforts have applied adaptive middleware solutions to satellite systems [3]. The initial prototype of the architecture presented in this paper is discussed in [4], and a description of how to use benefit functions is given in [5]. The middleware architecture presented in this paper is depicted in Fig. 1.

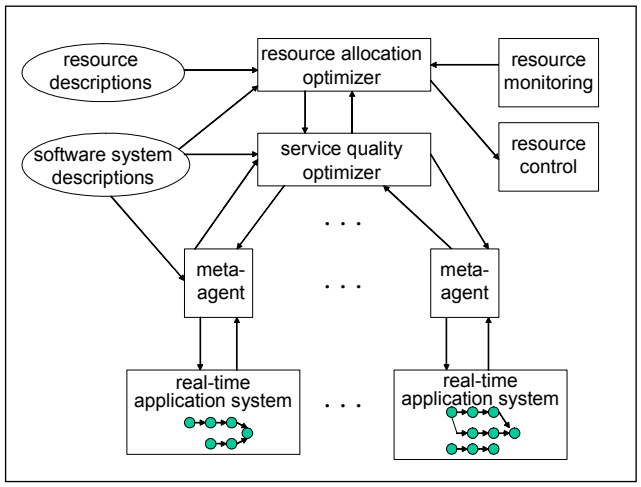

Fig 1. Architecture of the resource management middleware.

The **real-time application system** (RTAS) consists of one or more groups of tasks that have real-time constraints (these task groups are called paths). The tasks in a group may be distributed. Some tasks, task groups or systems may be able to run at varying qualities of service, depending on resource availability and on performance requirements. For example, at its highest quality of service, an image-processing algorithm would process every pixel in each image, and at a degraded quality of service it would ignore some pixels.

Each RTAS is managed by a **meta-agent** (MA), which monitors the real-time performance of the subsystem and determines when resource reallocations are necessary for maintaining adequate performance. To perform its task, an MA receives information on performance and resource needs from the RTAS and obtains data about each RTAS's relevant features (e.g., its structure and its real-time requirements). When a MA determines that its RTAS needs additional resources, it notifies the service quality optimizer.

The **service quality optimizer** (SQO) makes adjustments in quality-of-service (QoS) levels of one or more RTASs in order to satisfy requests from MAs. Upon receiving a request from an MA, the SQO determines if it is possible to satisfy the resource demands by adjusting QoS levels of RTASs. In addition to checking constraint feasibility, the SQO also considers the overall scientific benefit (called utility) that results from various QoS combinations, and selects the combination that yields the highest utility. Upon finding a suitable allocation, the SQO notifies the MAs of any QoS level changes that must be made. If the SQO cannot find a feasible allocation, it requests a reallocation of resources from the resource allocation optimizer.

The **resource allocation optimizer** (RAO) determines the best way to allocate resources to application systems. Its main task is to assign tasks to processors. To do this, it considers the utility that would result from various assignments and chooses the assignment with the highest utility. To make decisions about the suitability of allocations, the RAO obtains information from the **resource monitoring** component. The **resource control** component carries out reallocation decisions.

III. Usefulness For Onboard Processing In Satellite Systems

The resource management middleware has been applied to prototype software that is being developed under an ESTO-funded grant for possible use on future autonomous satellite systems The prototype contains software components that perform the following tasks: simulation of image capture by a camera, analysis of the captured images by an image processing agent (IPA), compression of images by a compression agent (CA), production of simulated health and safety information about the satellite by a health and safety data agent (HSDA), and download of data by the data management agent (DMA).

Resource management was applied to the Image Processing Subsystem, which contains a simulated camera sensor that collects image data, an agent (IPA) processes the data, and an actuator (CA) takes an action based on the results of processing the data. The entire processing "path" from camera through CA has a timing requirement.

The camera captures raw images of the Earth based on a schedule. The extrinsic attribute *observation_importance* specifies how important the area of observation is to the user. The extrinsic attribute *frequency* specifies the image capture rate. Both extrinsic attributes are reported to the Meta-Agent.

The Image Processing Agent (IPA) uses a simplified NOAA algorithm to calculate the cloud cover in an image to help establish the benefit of that image. The algorithm has been modified to skip pixels in order to speed up calculation, thus varying the amount of processing time required. The number of pixels processed is controlled by the service attribute *pixels* set by the Meta-Agent. The extrinsic attribute *cloud_cover* is reported to the Meta-Agent.

The Compressor Agent (CA) uses various compression algorithms on images presented to it by the IPA. Compression algorithms include *no compression, lossless compression, lossy compression*, and *discarding an image*. The compression algorithm, and thus resource usage, of the CA depends on the service attribute *compression*.

The Health and Safety Data Agent (HSDA) produces variable amounts of information about various metrics to help engineers on the ground quickly diagnose satellite problems. The amount of data the HSDA stores into the buffer is determined based on the service attributes: *power_service*, *pressure_service* and *temperature_service*. The HSDA passes the extrinsic attributes *power*, *temperature* and *pressure* to the Meta-Agent as measured from sensors on the satellite.

The Data Management Agent (DMA) keeps a record of all the data elements in the buffer and their associated download priorities. When the network downlink is available, the data

transfer process will connect to a data receiver and begin downloading the data in order by highest priority first. The priority of data is determined by the subsystem that produced the data and the extrinsic attributes related to the data such as the cloud-cover of an image. A confirmation of download for the data is sent to the satellite from the ground.

To better demonstrate the concept of operation for the prototype system under the control of the adaptive resource manager (ARM), consider the following example. Assume that all the ARM middleware is running on the system. When the camera, IPA, CA, HSDA, and DMA are started, the Allocation Manager Agent allocates agents to resources to maximize benefit and meet deadlines using default values for extrinsic attributes.

According to an *a priori* schedule, the camera reports the extrinsic attributes of *observation_importance* and *frequency* to the Meta-Agent, and takes an image of the Earth. The Meta-Agent sets the service attribute *pixels* for the IPA based on the available resources and the expected *cloud_cover*. The IPA processes the image and updates the Meta-Agents expectation of *cloud_cover*. Based on the available resources and the actual *cloud-cover*, the Meta-Agent sets the *compression* service attribute for the CA. The CA compresses the image and stores it in the buffer.

The resource manager has several options when choosing the *compression* service level. Based on user-defined benefit functions, a high value for the *cloud_cover* may indicate that the Meta-Agent should choose a high compression method since the image probably has little scientific benefit. This reduces the amount of space used in the buffer in order to make room for future images and health and safety data that may provide greater benefit. If there is little *cloud_cover* , the Meta-Agent notifies the Global Meta-Agent that the use of more buffer would produce more benefit than previously expected. The Global Meta-Agent evaluates the request for improving the service provided to the IPS, and may ask the Allocation Manager Agent for a new allocation or adjust the allowable service attributes for each sub-system based on overall system benefit. When the *cloud_cover* increases to a high value, the Meta-Agent reports lower benefit values received from running the sub-system with the current amount of resources available. The Global Meta-Agent may restrict the amount of resources that the sub-system is receiving by either asking for a reallocation or by altering the allowable service attributes through the Meta-Agent.

When the satellite is found to have normal extrinsic attribute values for *temperature*, *power* and *pressure*, the HSDA produces a normal or low amount of health and safety data. When *power*, *pressure* or *temperature* significantly deviates from the norm, the Meta-Agent sets the service attributes *power_service*, *temperature_service* or *pressure_service* to a higher value depending on what extrinsic attributes are abnormal. The degree to which the Meta-Agent will increase the service attribute depends on the amount of resources available for storing the data.

The interactions between sub-systems, such as the IPA and HSDA's competition for buffer space, can become quite complicated. Changing the service attributes associated with an agent may not adhere to monotonic properties in all resources. For example, changing the *compression* service attribute associated with the CA may increase processing requirements when increased but, at the same time, may decrease buffer requirements. The optimization framework must ensure that trade-offs between granting resources to one sub-system as opposed to another are weighed and evaluated in full to obtain optimality. This is a computationally prohibitive problem, but the use of a hierarchical agent-based architecture allows lower-level agents (Meta-Agents) to refine near-maximized benefit solutions proposed by higher-level agents (the Allocation Manager Agent and the Global Meta-Agent) to approach maximized benefit in real-time.

IV. Conclusions

One of the current trends in spacecraft software design is to increase the autonomy of onboard flight and science software. For many science missions, the ability of the spacecraft to autonomously respond in real-time to unpredicted science events is crucial for mission success. This paper presents resource management middleware that employs utility theory to optimize the real-time performance of application software and to achieve maximum system level benefit. It also illustrates how this methodology can be extended to manage both software and observational resources onboard a spacecraft to achieve the best possible observations.

References

[1] B. Ravindran, L. Welch and B. Shirazi, "Resource Management Middleware for Dynamic, Dependable Real-Time Systems," *The Journal of Real-Time Systems*, 20:183-196, Kluwer Academic Press, 2000.

[2] Lonnie R. Welch And Behrooz A. Shirazi, "A Dynamic Real-Time Benchmark for Assessment of QoS and Resource Management Technology," *The IEEE Real-Time Technology and Applications Symposium*, 36-45, June 1999.

[3] L. Welch, B. Pfarr, and B. Tjaden, "Adaptive Resource Management Technology for Satellite Constellations," *The Second Earth Science Technology Conference (ESTC-2002)*, Pasadena, CA, June 2002.

[4] S. Jain, L. Welch, D. Chelberg, Z. Tan, D. Fleeman, D. Parrott, B. Pfarr, M. Liu, and C. Shuler, "A collaborative problem solving agent for on-board real-time systems," *The 10th Workshop on Parallel and Distributed Real-Time Systems*, April 2002.

[5] D. Andrews, L. Welch, D. Chelberg, and S. Brandt, "A Framework for Using Benefit Functions in Complex Real-Time Systems," *The 10th Workshop on Parallel and Distributed Real-Time Systems*, April 2002.